REMARKS

Claims 1-25 and 27-47 are now pending in the application. Claims 1, 9, 16, 37, 39, 40, 41, and 42 have been amended. No new matter has been added. Reconsideration and reexamination are respectfully requested in view of the amendments and the following remarks.

Claim Rejections 35 USC § 103

The Examiner rejected claims 1-25 and 27-44 under 35 U.S.C. § 103(a) over Bristor (U.S. 6,018,342). The Examiner appears to have rejected claims 45-47, which were added in the previous response, on the same basis.

After the present amendment, each of the independent claims expressly recites the maintenance and use of "*a history of document states of a document.*" The applicant respectfully submits each of the claims is allowable over Bristor, at least in part and in conjunction with the applicant's previous arguments, because a "history of document states of a document" is *not* the same thing as a history of *commands* and is *not* the same thing as a list of *documents*. Rather, "a history of document states of a document" is – quite simply – a history of the states of a document.

The Examiner states that Bristor discloses maintaining in a memory a state history of a document. But Bristor provides a method for automatically organizing *commands ("user-generated signals")* – not document states. The histories that Bristor describes are not specific to a document, let alone previous states of a document. Thus, with Bristor's method, a user can re-execute a command, but the command is typically executed on the current state of documents. It cannot be used to recover a previous state of a document. In short, nothing in Bristor discloses or suggests maintaining a history of states of a document.

For example, a user of Bristor's system can save the URL of a visited website in a history list, and, at some later time, can select the URL from the history list and revisit the website – i.e. retrieve the document or documents making up the website. But rather than using the history list

to retrieve a previous state of the website, Bristor simply re-executes the saved command. As a result, Bristor retrieves the website as it currently exists – that is, in its current state at the time the command is re-executed; if the website has been changed or deleted, Bristor does not retrieve any previous state of the website. Thus, Bristor does not maintain a history of the states of the website.

Similarly, if a user of Bristor's system executes a shell command such as "list contents of directory," the command can be saved in a history list. At some later time, the user can select the command from the history list and re-execute it, causing the system to again list the contents of the directory. But Bristor's system will list the current contents of the directory – at the time the command is re-executed – and not the contents reflecting the previous state of the directory; once again, if the contents of the directory have been changed or if the directory has been deleted, the user will not be able to retrieve the previous contents of the directory. Thus, Bristor does not maintain a history of the states of the directory.

The Examiner recognizes that Bristor does not teach automatically capturing the state of the document as it exists after an operation that changes the state of the document and adding the captured state to the state history. But, more than that, Bristor fails to disclose or suggest maintaining any record of changes to the state of a document – indeed, Bristor's method is *insensitive* to changes in the state of documents. Thus, as the examples above suggest, a command that is re-executed from Bristor's history list is re-executed on the current state of its target (be it a website, directory or document), whether the user desires that or not. Bristor simply offers no means for keeping track of the changes in the state of the documents upon which commands are executed, and does not suggest that it might be desirable to track such changes.

The Examiner states that one skilled in the art would nevertheless be motivated to modify Bristor to capture the changed state of a document and add the captured state to a state history "because such a modification would allow a user in Bristor's system to create a draft ('snapshot') of the current state of the document at a particular point in time." But Bristor's focus is on providing an efficient way to keep track of user commands so that they can be re-executed, using

the current state of documents and directories. Bristor fails entirely to disclose or suggest both maintaining a history of document states of a document, and capturing a changed document state and adding the captured state to the history of document states.

The applicant recognizes that the rationale for combining or modifying references can be implied or reasoned from the prior art or from knowledge generally available to one of ordinary skill in the art. But the applicant submits that there is nothing in the Examiner's analysis to support that such a modification to Bristor might be desirable. In the absence of any other prior art or a convincing line of reasoning (*See Ex parte Clapp*, 227 USPQ 972, 973 (Bd. Pat. App. & Inter. 1985), the only basis for such a modification to Bristor is the hindsight provided by applicant's claims – and the use of hindsight to establish a prima facie case of obviousness is simply not proper (*Id.*; *see also* MPEP 2142, paragraph 2).

Because at least one element of each claim is not taught or suggested by Bristor , the applicant respectfully submits that no prima facie case of obviousness under 35 U.S.C. § 103 has been established. Accordingly, the applicant submits that all of the claims are in condition for allowance, which action is respectfully requested, in light of the above.
Please apply any charges or credits to Deposit Account No. 06-1050.

Respectfully submitted,

Date: 1/6/2003

Tamara Fraizer
Reg. No. 51,699

Fish & Richardson P.C.
2200 Sand Hill Road, Suite 100
Menlo Park, CA 94025
Telephone: (650) 322-5070
Facsimile: (650) 854-0875

Version with markings to show changes made

In the claims:

Claims 1, 9, 16, 37, 39, 40, 41 and 42 have been amended as follows:

1. (Thrice amended) A method implemented in a computer program application performing operations on documents having states, the method comprising:

maintaining in a memory a state history [of a document], the state history representing a history of document states of the document; and

whenever an interesting operation has occurred, an interesting operation being an operation by a user that changes the state of the document, automatically capturing the state of the document as it exists after the operation and adding the captured state to the state history of the document.

2. The method of claim 1, wherein the memory comprises a disk file.

3. The method of claim 1, further comprising:

maintaining in the state history the order in which the stored states were automatically added to the state history; and

displaying the state history to a user as a list of document states shown in their stored order.

4. The method of claim 3, wherein:

the list of document states displayed to the user comprises a list of items, each item representing a state of the document that existed after an interesting operation and that can be recovered directly by selecting the item.

5. The method of claim 4, further comprising:

providing a tool operable under user control to obtain source material from any state in the state history and apply it to a current state of the document, where the document is a raster image.

6. The method of claim 4, further comprising:

enabling a user to select any item in the displayed list of items and cause the application to create a new document having the document state corresponding to the selected item.

7. The method of claim 4, wherein:

each of the captured states in the state history maintains the state data in essentially its original form, whereby the captured state data is suitable for immediate use in other operations.

9. (Twice amended) A computer-implemented method of interacting with a user editing a document in a computer program application, the document having a document state, the method comprising:

receiving from the user a sequence of commands to change the document;

changing the document state in response to each command;

each time the document state is changed, adding the changed document state to a state history maintained in a computer-readable memory device, the state history representing a history of document states of the document [each time the document state is changed];

for each document state added to the state history, adding a corresponding entry to a history list displayed to the user on a computer-controlled display device operated as part of a graphical user interface; and

in response to a user action, selecting an item in the history list and establishing the document state corresponding to the selected item in the history list as the current state of the document.

10. The method of claim 9, wherein:

the state history and the history list are limited to storing a preset number of items and excess items are scrolled off the top of the list as new items are added.

11. The method of claim 9, wherein:

the state history is stored in a region of memory and the oldest document states in the state history are discarded when free space in the region runs low.

12. The method of claim 11, wherein:

the oldest document states are found and discarded by a memory management process.

13. The method of claim 9, further comprising:

in response to a user command to change the document state corresponding to the selected item in the history list and established as the current state of the document, deleting the items after the selected item in the history list and the corresponding document states from the state history.

14. The method of claim 9, further comprising:

in response to a user command to change the document state corresponding to the selected item in the history list and established as the current state of the document, maintaining the items after the selected item in the history list and adding a new item to the end of the history list and a new document state to the state history.

15. The method of claim 9, further comprising:

enabling a user interface gesture on the history list to create a new document from a document state from the state history.

16. (Twice Amended) A method implemented in a computer program application operable to create and edit a document, comprising:

maintaining in a memory a state history [of a document], the state history representing a history of document states of the document;

in response to a user action, selecting a first state from the state history and establishing the first selected state of the document as the current state of the document;in response to a user action, selecting a second state from the state history, the second state being a state created after

the first state, as a source of data for an operation; andperforming the operation with the data from the second state on the first state.

17. A method implemented in a computer program application operable to create and edit a document, comprising:

keeping a history of document states of a document, the document states being created automatically whenever a user command to the application changes the state of the document and being complete in themselves;

enabling the user to discard any of the states in the history to create a revised history; and

enabling the user to step backward and forward through the revised history and thereby alter the state of the document to be any of the document states in the revised history.

18. A method implemented in a computer program application operable to create and edit a document, comprising:

keeping a history of document states of a document, the document states being created automatically whenever a user command to the application changes the state of the document and being complete in themselves;

enabling the user to discard any of the states in the history to create a revised history; and

enabling the user to designate any one of the document states in the revised history and thereby establish the designated state as the current state of the document.

19. The method of claim 18, further comprising:

saving the history when the document is closed on a long-term storage medium, whereby the history may be restored when the document is later opened and across invocations of the application.

20. The method of claim 19, wherein:

the saved history resides in the document with final document data.

21. The method of claim 19, wherein:

the saved history resides in a long-term data repository independent of the original document.

22. A method enabling a user to control operation of a computer program application for creating and modifying a document, the method comprising:

identifying for the user on a display device a set of states that the document has been in by operation of the application; and

enabling the user to designate any one of the identified states as a document state operand.

23. The method of claim 22, further comprising:

displaying the document in a user interface window, the document being a digital image.

24. The method of claim 23, wherein the digital image has a plurality of layers, each of the plurality of layers having a plurality of channels, the method further comprising:

displaying user-interface elements for applying filters to the digital image.

25. The method of claim 22, further comprising:

establishing the designated state as the current state of the document in response to a user command.

27. The method of claim 22, further comprising:

providing the user a delete tool for deleting the designated state from the set of states.

28. The method of claim 22, wherein:

the set of states is identified by displaying a scrollable list of elements each identifying one of the states in the set.

29. The method of claim 28 wherein the list elements are ordered by the time the corresponding states were created.

30. The method of claim 25 wherein the designation and establishment are performed in response to a single command.

31. The method of claim 25 wherein the set of states is displayed in an order, the method further comprising:

enabling the user to make a gesture on a user interface indicating a sequence of displayed state identifiers and responding to the gesture by displaying the document in the states indicated as the gesture is made.

32. The method of claim 25 further comprising:

enabling the user to modify the document state after the establishing step; and

adding the document state resulting from the modification to the set of states identified on the display device.

33. The method of claim 31 wherein the set of states is displayed in order of creation of the states in the set.

34. The method of claim 31 wherein the document is a digital image.

35. The method of claim 25 further comprising:

providing a step backward and a step forward command for the user to execute to navigate the set of states; and

providing a separate undo and redo command for the user to undo and redo commands entered by the user.

36. The method of claim 22, further comprising:

providing a step backward and a step forward command for the user to execute to navigate the set of states; and

providing a separate undo and redo command for the user to undo and redo commands entered by the user.

37. (Thrice amended) Apparatus comprising a computer-readable storage medium tangibly embodying program instructions defining a computer program application for performing operations on documents having states, the program comprising instructions operable for causing a programmable processor to:

maintain in a memory a state history [of a document], the state history representing a history of document states of the document; and

whenever an interesting operation has occurred, an interesting operation being an operation by a user that changes the state of the document, the state being complete in itself and independent of other states; automatically capture the state of the document as it exists after the operation and add the captured state to the state history of the document.

39. (Twice amended) Apparatus comprising a computer-readable storage medium tangibly embodying program instructions for interacting with a user editing a document in a computer program application, the document having a document state, the apparatus comprising instructions operable for causing a programmable processor to:

receive from the user a sequence of commands to change the document;

change the document state in response to each command;

each time the document state is changed, add the changed document state to a state history maintained in a computer-readable memory device, the state history representing a history of document states of the document [each time the document state is changed];

for each document state added to the state history, add a corresponding entry to a history list displayed to the user on a computer-controlled display device operated as part of a graphical user interface; and

in response to a user action, select an item in the history list and establish the document state corresponding to the selected item in the history list as the current state of the document.

40. (Twice Amended) A computer program, residing on a computer-readable medium, comprising instructions for causing a computer to:

keep a state history [list], the state history representing a history of document states of the document;

in response to a user action, select a first state from the state history [list] and establish the first selected state of the document as the current state of the document;

in response to a user action, select a second state from the state history [list], the second state being a state created after the first state, as a source of data for an operation; and

perform the operation with the data from the second state on the first state.

41. (Twice Amended) A computer program, residing on a computer-readable medium, comprising instructions for causing a computer to:

keep a history of document states [created by a user] of a document; the document states being created automatically whenever a user command to the application changes the state of a document and being complete in themselves;

enable the user to discard any of the states in the history to create a revised history; and

enable the user to step backward and forward through the revised history and thereby alter the state of the document to be any of the document states in the revised history.

42. (Twice Amended) A computer program, residing on a computer-readable medium, comprising instructions for causing a computer to:

keep a history of document states [created by a user] of a document, the document states being created automatically whenever a user command to the application changes the state of a document and being complete in themselves;

enable the user to discard any of the states in the history to create a revised history; and

enable the user to designate any one of the document states in the revised history and thereby establish the designated state as the current state of the document.

43. A computer program, residing on a computer-readable medium, comprising instructions for causing a computer to:

create and modify a document;

identify for a user on a display device a set of states that the document has been in by operation of the application; and

enable the user to designate any one of the identified states.

44. The method of claim 36, further comprising:

providing to the user a first undo command function that operates with reference to the first history and a second undo command function that operates with reference to the second history.

45. The method of claim 3, further comprising:

establishing as the current state of the document a state stored in the state history.

46. The method of claim 1, further comprising:

maintaining in memory a history of all operations requested by a user, including operations global to the state of the application.

47. The apparatus of claim 37, further comprising instructions operable for causing a programmable processor to:

maintain in memory a history of all operations requested by a user, including operations global to the state of the application.